ISSN 0265-2919

80p

26

THE HOME COMPUTER ADVANCED COURSE

MAKING THE MOST OF YOUR MICRO

An ORBIS Publication

IR £1 Aus $1.95 NZ $2.25 SA R1.95 Sing $4.50 USA & Can $1.95

CONTENTS

APPLICATION

NEW MUSICAL EXPRESSIONS We look at the development of the sequencer 509

HARDWARE

ONE STEP BEYOND How well does the Sinclair QL measure up to expectations? 501

SOFTWARE

BLACK MAGIC Necromancer is a game demanding skill and fast reactions 517

COMPUTER SCIENCE

ADVANCE TO LOGO The start of a new language series 506

JARGON

EAPROM TO ELECTROSENSITIVE A weekly glossary of computing terms 508

PROGRAMMING PROJECTS

SWINGTIME A program to play Hangman 504

PROGRAMMING TECHNIQUES

THE HUMAN FACTOR We look at the importance of user-friendly program design 512

MACHINE CODE

HIGHLY PROCESSED CODE The first in a new course for Dragon and Tandy owners 518

PROFILE

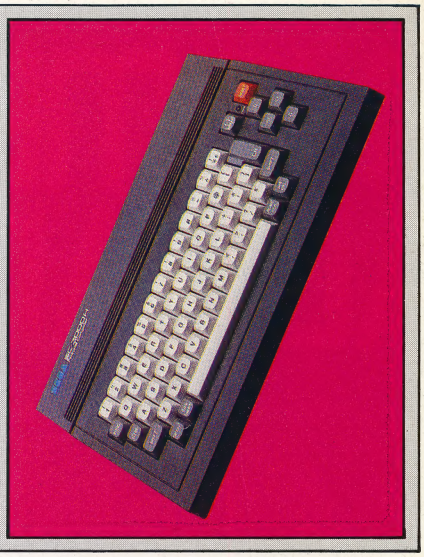
NEW HORIZONS Psion are a versatile and forward-looking team 520

WORKSHOP

HOT WIRED We relaunch our practical series with a new project 514

Next Week

- **We review the SEGA SC 3000, a colourful new Japanese computer with a wealth of manufacturer's software.**
- **Having made up the user port lead, we begin to build the BBC Micro/Commodore 64 universal interface.**
- **Ergonomics in design engineering has gradually changed the look and functions of offices, houses and factories: we examine its impact on computers.**



QUIZ

1) What is the infamous 'kludge'?

2) What does IDC stand for?

3) What is the advantage of digitising a sound, such as a drumbeat?

4) What is an emulator?

Answers To Last Week's Quiz

A1) Fillsub's parameters are the start-point co-ordinates, and the minimum and maximum X co-ordinates.

A2) MUsic Simulator Interpreter for COMpositional Procedures was designed by Lejaren Hiller to aid musical composition on computers.

A3) The central character of Lords of Midnight is Luxor the Moonprince, Lord of the Free.

A4) The programming technique, 'firewalling', is the process of checking parameters as they pass between routines.

QUIZ

CLUBSPOT A guide to the micro user's Prestel pages INSIDE BACK COVER

Editor Jim Lennox; **Managing Editor** Mike Wesley; **Art Director** David Whelan; **Technical Editor** Brian Morris; **Production Editor** Catherine Cardwell; **Art Editor** Claudia Zeff; **Chief Sub Editor** Robert Pickering; **Designer** Julian Dorr; **Art Assistant** Liz Dixon; **Editorial Assistant** Stephen Malone; **Sub Editor** Steve Mann; **Researcher** Melanie Davis; **Contributors** Andrew Stark, Geoff Bains, Harvey Mellor, Martin Wheatcroft, Mike Curtis,Steve Colwill, Steve Malone, Rory Forsyth, Jim Lennox, Richard Pawson, Graham Storrs; **Group Art Director** Perry Neville; **Managing Director** Stephen England; **Published by Orbis Publishing Ltd: Editorial Director** Brian Innes; **Project Development** Peter Brookesmith; **Executive Editor** Chris Cooper; **Production Controller** Peter Taylor-Medhurst; **Circulation Director** David Breed; **Marketing Director** Michael Joyce; **Designed and produced by Bunch Partworks Ltd; Editorial Office** 14 Rathbone Place, London W1P 1DE; **© APSIF Copenhagen 1984; © Orbis Publishing Ltd 1984: Typeset by Universe; Reproduction by Mullis Morgan Ltd; Printed in Great Britain by Artisan Press Ltd, Leicester**

HOME COMPUTER ADVANCED COURSE – Price UK 80p IR £1.00 AUS $1.95 NZ $2.25 SA R1.95 SINGAPORE $4.50 USA and CANADA $1.95

How to obtain your copies of HOME COMPUTER ADVANCED COURSE – Copies are obtainable by placing a regular order at your newsagent, or by taking out a subscription. Subscription rates: for six months (26 issues) £23.80; for one year (52 issues) £47.60. Send your order and remittance to Punch Subscription Services, Watling Street, Bletchley, Milton Keynes, Bucks MK2 2BW, being sure to state the number of the first issue required.

Back Numbers UK and Eire – Back numbers are obtainable from your newsagent or from HOME COMPUTER ADVANCED COURSE. Back numbers, Orbis Publishing Limited, 20/22 Bedfordbury, LONDON WC2N 4BT at cover price. AUSTRALIA: Back numbers are obtainable from HOME COMPUTER ADVANCED COURSE. Back numbers, Gordon & Gotch (Aus) Ltd, 114 William Street, PO Box 767G Melbourne, Vic 3001. SOUTH AFRICA, NEW ZEALAND, EUROPE & MALTA: Back numbers are available at cover price from your newsagent. In case of difficulty write to the address in your country given for binders. South African readers should add sales tax.

How to obtain binders for HOME COMPUTER ADVANCED COURSE – UK and Eire: Please send £3.95 per binder if you do not wish to take advantage of our special offer detailed in Issues 5, 6 and 7. EUROPE: Write with remittance of £5.00 per binder (incl. p&p) payable to Orbis Publishing Limited, 20/22 Bedfordbury, LONDON WC2N 4BT. MALTA: Binders are obtainable through your local newsagent price £3.95. In case of difficulty write to HOME COMPUTER ADVANCED COURSE BINDERS, Miller (Malta) Ltd, M.A. Vassalli Street, Valletta, Malta. AUSTRALIA: For details of how to obtain your binders see inserts in early issues or write to HOME COMPUTER ADVANCED COURSE BINDERS, First Post Pty Ltd, 23 Chandos Street, St. Leonards, NSW 2065. The binders supplied are those illustrated in the magazine. NEW ZEALAND: Binders are available through your local newsagent or from HOME COMPUTER ADVANCED COURSE BINDERS, Gordon & Gotch (NZ) Ltd, PO Box 1595, Wellington. SOUTH AFRICA: Binders are available through any branch of Central Newsagency. In case of difficulty write to HOME COMPUTER ADVANCED COURSE BINDERS, Intermag, PO Box 57394, Springfield 2137.

Note – Binders and back numbers are obtainable subject to availability of stocks. Whilst every attempt is made to keep the price of the issues and binders constant, the publishers reserve the right to increase the stated prices at any time when circumstances dictate. Binders depicted in this publication are those produced for the UK market only and may not necessarily be identical to binders produced for sale outside the UK. Binders and issues may be subject to import duty and/or local taxes, which are not included in the above prices unless stated.

COVER PHOTOGRAPHY BY IAN McKINNELL

ONE STEP BEYOND

IAN McKINNELL

Meet The Family
The Sinclair name has appeared on a considerable range of products: from audio amplifiers and the celebrated Black Watch, through calculators and computers, to the long-awaited flat-screen television and electric car. Technical ingenuity and innovation, stylish hi-tech product design, and ambitious marketing strategies have been Sinclair hallmarks from the start, though critics would say that press-release engineering, gimmickry and over-optimistic delivery schedules were more accurate labels

When Clive Sinclair announced the ZX Spectrum, he claimed that it was comparable with the BBC Micro, which cost twice as much. Two years later, when announcing the QL, he was even more ambitious, and compared the QL with computers costing five times as much, such as the IBM PC and the Apple Macintosh.

It is little wonder that so many people are confused about the Sinclair QL and its real worth. Firstly there was the massive media hype of the launch. The public were told that the QL was the most advanced home computer to date, boasting a 32-bit 68008 CPU, a four-program suite of integrated business software and a resident BASIC that would surpass any version available.

However the excitement provoked by the QL's announcement gradually wore off, as delivery dates approached and receded, and people realised that perhaps the claims were slightly exaggerated. A real backlash came when it was realised — Sinclair's assurances notwithstanding — that the machine was nowhere near ready for public release: all the money paid by the advance mail-order customers was just going to sit in the Sinclair bank accounts earning interest, while the anxious public waited and recalled previous Sinclair machine launches and deliveries.

There was, however, one big difference between the Spectrum and QL launches — although the Spectrum 28-day delivery promises were not met, nonetheless review models of the machine were given to journalists immediately after the launch, the machines did work, and they were virtually identical to those eventually released to the public; in the case of the QL, however, Sinclair had to admit that the promised 'SuperBASIC' and the operating system would not fit into the allocated 32 Kbytes of ROM — 48 Kbytes were going to be needed, but there was no room on the circuit board for the extra chip!

Rather than waste time and money redesigning the circuit board, the Sinclair staff came up with the now-infamous 'kludge' (sometimes wrongly called the 'dongle'). This was a small black plastic box protruding from the QL's cartridge port and containing the missing parts of BASIC and the operating system.

This at least enabled Sinclair to get some working machines out of the factory door, with the promise that the machines would be 'de-kludged' by a true upgrade later. The original 28-day delivery claims had by now turned into three months. Several versions of the operating system

went out in quick succession, each with its own flaws. Sinclair finally settled on one called the 'AH', and this has become the first version to be released in volume.

Consumer reaction to the kludge was not favourable — it was a visible proof of the machine's inadequacy, and hardly an advertisement for reliability. To get round this, Sinclair came up with an idea guaranteed to set any electronic engineer's teeth on edge — as there was no socket on the circuit board for the third 16 Kbyte ROM chip, it was placed 'piggyback' fashion on top of the existing chip, and all but one of the chip's 28 legs were individually soldered onto those of the chip below. The last leg was connected by a flying wire to another part of the board, so that the new chip could be accessed independently of its host. This meant that the kludge could be removed, but nothing had really changed.

All the early machines used EPROMs rather than ROMs, saving Sinclair the time it takes to produce ROMs but costing the company money — these 16 Kbyte EPROMs were selling at £80 each, and each QL used three of them. Although Sinclair would have bought at discount, the price must still have been a desperately high fraction of the machine's price, thus possibly forcing Sinclair into an early acceptance of the AH version operating system, despite its various bugs. Once it was settled upon, the ROMs to replace costly EPROMs could be manufactured and installed, and the company could hope to start turning a profit on the machine. Unfortunately, a debugged AH version is unlikely to appear before 1985.

With this in mind, how can we assess the QL? It can be seen as two machines in one: a powerful home computer, or a modest business machine, and, as such, it can genuinely claim to be a pioneer. It is probably closer to the conventional idea of the home micro, however: it is small, can produce a television display, has a resident BASIC, is sold by mail order (and by the high street shops soon), and has high-resolution colour graphics and joystick ports. Two features support its claims to business machine status: the built-in Microdrives provide reasonably substantial mass-storage (certainly by comparison with cassettes), and the machine comes 'bundled' with four applications programs — word processor, spreadsheet, database and graphics support.

As a home micro, the QL looks like very good value, given the Microdrives and the software, especially since it is of good quality, and since home users rarely see databases or spreadsheets. On the other hand, most home users don't need these applications, and don't have a use for them. The typical home user likes playing games and writing BASIC programs, which is fine in the latter case, since SuperBASIC is certainly one of the best dialects yet produced. There is an obvious shortage of commercial software for the QL, given the problems with producing the machine. Software production is not helped by the new Microdrives' incompatibility with the Spectrum version, by the QL screen's having a different layout from the Spectrum's, and by the operating system differences and problems.

There are, apparently, still bugs in SuperBASIC, and the editor is very similar to the line editor used on the Spectrum; this is not a bug, of course, but it isn't exactly Fifth Generation standard, either. The addition of BBC-like procedures and functions, and a SELECT structure similar to PASCAL's CASE are very real improvements, though the ON ERROR command is not implemented. The graphics are very good, and the sound is at least audible, though otherwise disappointing.

As a business machine, the QL is less

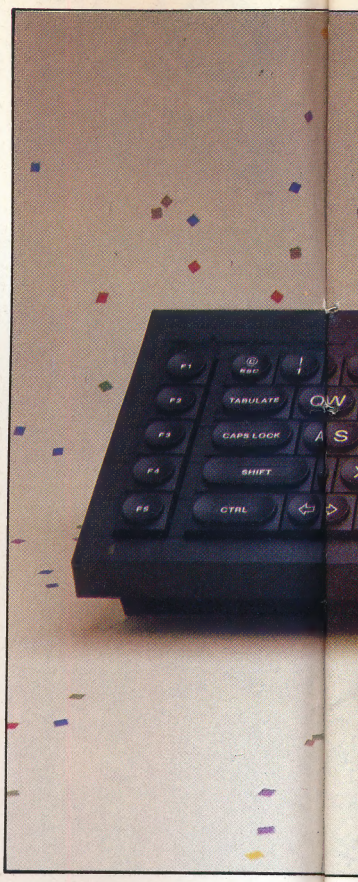
The Competition

QL vs BBC Micro

To claim the serious home user market, the QL must compete with the BBC Micro. In the QL's favour, the BBC Micro does not include free software, mass storage, a modern microprocessor, state-of-the-art graphics or a large expandable memory; on the other hand, the QL does not have government approval for school use, nor an enormous user base in homes, schools and (to a lesser extent) business, a huge range of high-quality third-party peripherals, a vast software catalogue (much of it free), or second-processor/upgrade capability

QL vs Macintosh

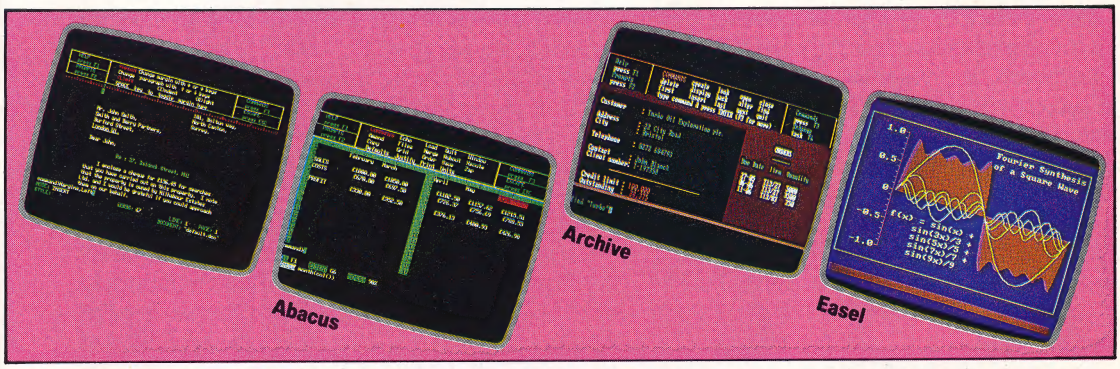
Sir Clive Sinclair timed the introduction of the QL to steal the thunder from the impending introduction of the Macintosh. In doing so, and in designing a machine around the same microprocessor, Sinclair has pitted the QL directly against the Mac, a machine four to five times its price. Although the two machines share many features, including memory size, clock speed of the CPU, and integrated software, there really is no comparison between the QL and Macintosh. The QL is a technically brilliant machine, but there is nothing new in the way it operates. Apple's Macintosh, on the other hand, boasts an operating system with icons, windows, and the ingenious mouse that sets the machine apart from every other computer on the market. The Macintosh represents a quantum leap forward and sideways — the forerunner of a new generation of microcomputers



Integrated Software

All four packages have similar screen formats and commands, and clear, concise displays. Data can be moved between them via the Microdrive. All four programs have bugs in this initial version, such as non-functioning commands and input/output errors, but these should be easy to fix in subsequent versions. Quill, a word processor, allows for 40-, 64-, or 80-character displays; typed characters are slow to appear on screen, which can be irritating. Abacus is an innovative spreadsheet with many built-in functions and the ability to label and address a group of ceils, but the 15K workspace left is virtually useless. Archive is a database with built-in commands for simple filing tasks. It can also be programmed via an internal language similar to SuperBASIC, but the Microdrives make it very slow. Easel creates bar graphs, pie charts, and line graphs of numeric data, and can switch rapidly between formats

IAN McKINNELL



IAN McKINNELL

SINCLAIR QL

PRICE
£399 (inc. VAT)

DIMENSIONS
472×138×46 mm

CPU
Motorola 68008, 7.5 MHz

MEMORY
128K RAM (expandable to 640K), 48K ROM

SCREEN
25 lines of 85 characters (with monitor); high resolution graphics: 512×256 pixels (4 colours), 256×256 (8 colours)

INTERFACES
Serial RS232 (2), Joysticks (2), Microdrives, LAN, TV, RGB monitor

LANGUAGES AVAILABLE
SuperBASIC

KEYBOARD
Pseudo-typewriter-style; 65 keys, including true space bar and five function keys, but no delete key

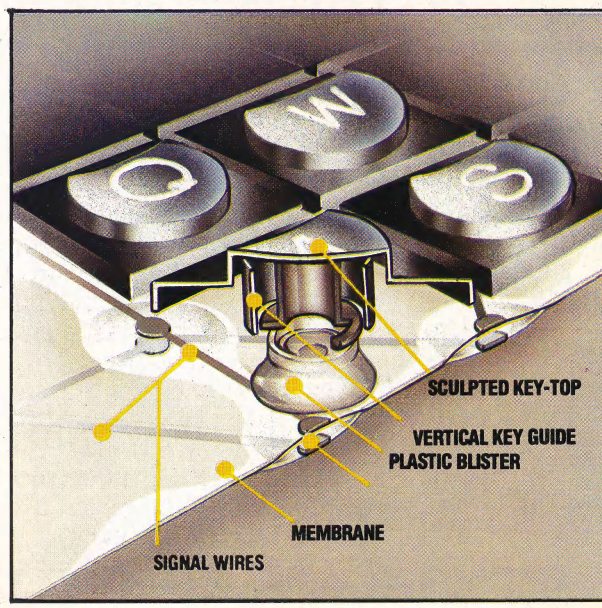
DOCUMENTATION
The user manual (in a ring binder) is of a high standard, and includes manuals for SuperBASIC and the applications software

STRENGTHS
Very fast number-crunching 68008 CPU, high-quality software included, very good graphics, advanced BASIC dialect

WEAKNESSES
Built-in Microdrives slower than disks, not compatible with Spectrum, little software available, operating system not fully de-bugged

convincing: the bundled software is good value, but any business user will surely want an accounts package at the least, and the spreadsheet program leaves only 15 Kbytes of user RAM, which rules out most serious financial models. The speed and questionable reliability of the Microdrives calls the whole QL mass-storage capability into question, especially as there is no disk drive interface. The keyboard seems unlikely to withstand heavy daily use, and it's difficult to imagine proficient typists accepting its peculiarities. The lack of commercial software is even more of a drawback for the machine's business users, and that plus the mass-storage deficiencies seem likely to end the machine's business career before it begins.

Like all Sinclair products the QL is exciting, innovative, controversial and occasionally frustrating. Though it cannot fairly claim to meet any of its targets, the QL has given its competitors a new standard to meet, and a new benchmark for comparison.



Bubble Memory

The QL has a 'membrane'-style keyboard. Two signal wires are held apart by a bubble in a plastic membrane, and when the key is pressed, it squashes the bubble and the wires make contact. Resistance and return force are provided by the plastic 'blister'. With sculpted full-travel plastic keys in a flat near-standard QWERTY layout, the QL keyboard is a big improvement on the Spectrum's, but its flimsy construction, and vague 'spongy' feel make it a poor match for the Vic-20 — never mind the Macintosh or the BBC Micro

ROSALIND BUCKLAND

SWINGTIME

KEVIN JONES

Hangman is a traditional wordgame that is easily implemented on home computers and can prove educational. Programming a Hangman game provides us with an opportunity to explore string manipulation. We discuss the structure of a simple version of the game, and give listings for the BBC Micro and the Spectrum.

Probably everybody has played a game of Hangman at some time. The object of the game is simply to guess the letters in a word. The only information given is the number of characters in the word, all of which are represented by dashes. A correctly guessed letter is displayed in its proper position, and an incorrect guess causes a part of a picture of a man on a scaffold to be drawn. In our program, there are 10 parts to this drawing, which is shown on the right-hand side of the screen. If all 10 parts are completed before you've filled in all the letters in the word, then the man is hanged and you've lost the game.

The basic principle of our program is very simple. It involves checking to see if a letter entered at the keyboard is contained in a randomly selected 'secret' word (one of 11 words held in DATA statements at the end of the program). If the guessed letter is present, it is displayed on the screen in its correct place. If the letter is not present, the program must display it anyway, to remind the player that it has already been used. In addition, of course, the program must then be made to jump to a subroutine that will draw a part of the hanged man.

The words that our programs use are stored in lines 1020 and 1030 in both versions. There is no reason why you can't add to these, using more DATA statements. But if you do add your own lexical brain-teasers, you must remember that they should not be more than 10 letters long. (Although this restriction could be lifted by altering lines 30 and 50.) Also, the total number of words in the DATA statements must be added up, and the value of N in line 20 altered accordingly.

At the beginning of the program, all of the words are read into an array. One of the elements of the array is picked at random, and a line of dashes corresponding to the length of the word is displayed on the screen. The rest of the game consists of a repetitive loop. When a letter is entered from the keyboard it is screened: if it is more than one letter, or not a character at all, then the program makes a BEEP and loops back for another input. The letter is also checked against the list of letters that have already been tried in the game. If it has been used before, then a warning is flashed on the screen and a new letter read.

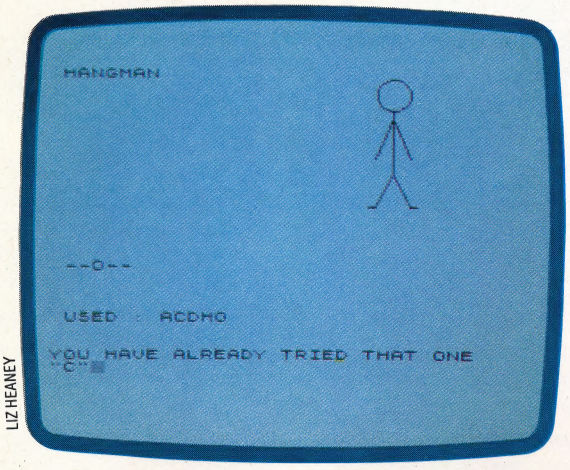
If the trial letter is acceptable, it is added to the displayed list of used characters, and then compared against each letter of the word in turn. If it matches in any position, it is put up on the screen in place of the corresponding dash. If no match is found in the whole word, a subroutine is called to draw a part of the hanging scene.

If the player has had 10 wrong guesses then the man is hanged, a short consolatory tune is played and a new word is selected. Alternatively, if all the letters of the word have been correctly placed, a congratulatory musical phrase is played.

Our two versions of the program can be easily adapted to most home micros. The subroutines to draw the man and the scaffold, of course, need to be especially adapted to the graphics capabilities of individual machines. Programmers with an interest in creating interesting screen displays may like to elaborate on the drawn result: a hanged man gently swinging in the breeze if the player fails to win, perhaps?

Other refinements can also be added to the game. We suggest that it may be a good idea to add a check at the beginning of the program to see that a selected word has not already been used. As the program stands, the selection of the word is a purely random matter, and the same word could be chosen twice in succession.

Even more helpful would be a routine to screen the inputs so that only upper case letters were accepted as trial letters. (The program as given will accept any character as a trial 'letter'.) Lower case letters, numbers, and symbols could be checked by their ASCII codes and rejected. As it is given here, however, our program does play a good game of Hangman, and offers opportunities for adventurous programmers.



LIZ HEANEY

Progress Report
The game in progress, showing the development of the hanged man and the progression of the word

BBC Micro

```
                Hangman

10 REM
20 N=11
30 DIM W$(N)
40 REM initialise
50 D$="----------"
60 FOR I=1 TO N
70   READ W$
80   W$(I)=W$
90   NEXT I
100 MODE 1
110 REM start new word
120 CLS
130 PRINT TAB(5,1);"HANGMAN"
140 W$=W$(RND(N))
150 C=0:W=0:D=0
160 G$=""
170 PRINT TAB(5,23);"Letters tried :"
180 L=LEN(W$)
190 PRINT TAB(5,20);LEFT$(D$,L)
200 REM input the trial letter
210 PRINT TAB(5,30);STRING$(34," ")
220 INPUT TAB(5,30);L$
230 REM check input letter
240 IF L$="" THEN SOUND 1,-15,50,5:GOTO 210
250 IF LEN(L$)>1 THEN SOUND 1,-15,50,5:GOTO 210
260 F=0
270 FOR I=1 TO LEN(G$)
280   IF L$<>MID$(G$,I,1) THEN GOTO 330
290   PRINT TAB(5,30)"YOU HAVE ALREADY TRIED THAT LETTER"
300   SOUND 1,-15,50,5
310   T%=TIME:REPEAT UNTIL TIME>T%+50
320   F=1
330   NEXT :
340 IF F=1 THEN 210
350 G$=G$+L$
360 PRINT TAB(5,25);G$
370 REM test letter against word
380 F=0
390 FOR I=1 TO L
400   IF L$=MID$(W$,I,1) THEN GOSUB 480
410   NEXT I
420 IF F<>1 THEN GOSUB 670
430 IF D=1 THEN GOSUB 590:GOTO 120
440 IF C=L THEN GOSUB 530:GOTO 120
450 GOTO 210
460 END
470 REM letter matches
480 PRINT TAB(4+I,20);L$
490 C=C+1
500 F=1
510 SOUND 1,-15,200,5:SOUND 1,0,0,2
520 RETURN
530 REM success!
540 FOR I=1 TO 200 STEP 10
550   SOUND 1,-15,I,2
560   NEXT I
570 RETURN
580 REM failure!
590 FOR I=200 TO 1 STEP -10
600   SOUND 1,-15,I,2
610   NEXT I
620 CLS
630 PRINT TAB(5,20);"THE WORD WAS : ";W$
640 T%=TIME:REPEAT UNTIL TIME>T%+200
650 RETURN
660 REM letter doesn't match
670 W=W+1
680 ON W GOTO 690,760,780,810,830,870,890,910,930,960
690 VDU29,300;800;
700 MOVE 50,0
710 FOR A=0 TO 7 STEP 0.4
720   DRAW 50*COS(A),50*SIN(A)
730   NEXT A
740 VDU29,0;0;
750 RETURN
760 MOVE 800,750:DRAW 800,550
770 RETURN
780 MOVE 740,450:DRAW 750,450:DRAW 800,550
790 DRAW 850,450:DRAW 860,450
800 RETURN
810 MOVE 750,630:DRAW 800,730:DRAW 850,630
820 RETURN
830 PLOT69,800,800
840 PLOT69,820,820
850 PLOT69,780,820
860 RETURN
870 MOVE 780,790:DRAW 800,770:DRAW 820,790
880 RETURN
890 MOVE 600,400:DRAW 1100,400
900 RETURN
910 MOVE 1000,400:DRAW 1000,900
920 RETURN
930 MOVE 1000,900:DRAW 800,900
940 MOVE 1000,850:DRAW 950,900
950 RETURN
960 MOVE 800,900:DRAW 800,850
970 MOVE 780,790:PLOT 14,800,770:PLOT 6,820,790
980 MOVE 780,770:DRAW 800,730:DRAW 820,770
990 D=1
1000 RETURN
1010 REM the words
1020 DATA "MUTATION","REBATE","SERAPH","HANGMAN","BBC"
1030 DATA "SPECTRUM","GEOFF","ELEPHANT","XENOPHOBE",
     "WRINKLE","GRENADE"
```

Spectrum

```
HANGMAN

10 REM
20 LET N=11
30 DIM X$(N,10)
35 DIM Y(N)
40 REM initialise
50 LET D$="----------"
60 FOR I=1 TO N
70 READ W$
75 LET Y(I)=LEN W$
80 LET X$(I)=W$
90 NEXT I
110 REM start a new word
120 CLS
130 PRINT AT 1,1;"HANGMAN"
140 LET I=(1+INT (RND*N))
145 LET W$=X$(I)
150 LET C=0: LET W=0: LET D=0
160 LET G$=""
170 PRINT AT 20,1;"USED : "
180 LET L=Y(I)
190 PRINT AT 16,1;D$(1 TO L)
200 REM input the trial letter
220 INPUT L$
230 REM check input letter
240 IF L$="" THEN  BEEP 0.25,-10: GO TO 210
250 IF LEN L$>1 THEN  BEEP 0.25,-10: GO TO 210
260 LET F=0
270 FOR I=1 TO LEN G$
280 IF L$<>G$(I) THEN  GO TO 330
290 PRINT AT 21,1;"YOU HAVE ALREADY TRIED THAT ONE"
300 BEEP 0.25,-10
310 PAUSE 25: PRINT AT 21,1,,
320 LET F=1
330 NEXT I
340 IF F=1 THEN  GO TO 210
350 LET G$=G$+L$
360 PRINT AT 20,8;G$
370 REM test letter against word
380 LET F=0
390 FOR I=1 TO L
400 IF L$=W$(I) THEN  GO SUB 480
410 NEXT I
420 IF F<>1 THEN  GO SUB 670
430 IF D=1 THEN  GO SUB 590: GO TO 120
440 IF C=L THEN  GO SUB 530: GO TO 120
450 GO TO 210
460 STOP
470 REM letter matches
480 PRINT AT 16,I;L$
490 LET C=C+1
500 LET F=1
510 BEEP 0.25,+16
520 RETURN
530 REM success!
540 FOR I=-10 TO 10
550 BEEP 0.1,I
560 NEXT I
570 RETURN
580 REM failure!
590 FOR I=10 TO -10 STEP -1
600 BEEP 0.1,I
610 NEXT I
620 CLS
630 PRINT AT 10,3;"THE WORD WAS : ";W$
640 PAUSE 50
650 RETURN
660 REM letter doesn't match
670 LET W=W+1
680 IF W=1 THEN  GO TO 690
681 IF W=2 THEN  GO TO 760
682 IF W=3 THEN  GO TO 780
683 IF W=4 THEN  GO TO 810
684 IF W=5 THEN  GO TO 830
685 IF W=6 THEN  GO TO 870
686 IF W=7 THEN  GO TO 890
687 IF W=8 THEN  GO TO 910
688 IF W=9 THEN  GO TO 930
689 IF W=10 THEN  GO TO 960
690 CIRCLE 200,150,10
700 RETURN
760 PLOT 200,140: DRAW 0,-40
770 RETURN
780 DRAW -10,-20: DRAW -4,0
790 PLOT 200,100: DRAW 10,-20: DRAW 4,0
800 RETURN
810 PLOT 190,110: DRAW 10.25: DRAW 10,-25
820 RETURN
830 PLOT 200,150
840 PLOT 196,155
850 PLOT 204,155
860 RETURN
870 PLOT 196,148: DRAW 4,-4: DRAW 4,4
880 RETURN
890 PLOT 170,60: DRAW 80,0
900 RETURN
910 PLOT 240,60: DRAW 0,115
920 RETURN
930 DRAW -40,0
940 PLOT 240,165: DRAW -10,10
950 RETURN
960 PLOT 200,175: DRAW 0,-15
970 OVER 1: PLOT 196,148: DRAW 4,-4: DRAW 4,4: OVER 0
980 PLOT 196,144: DRAW 4,4: DRAW 4,-4
990 LET D=1
1000 RETURN
1010 REM the words
1020 DATA "MUTATION","REBATE","SERAPH","HANGMAN","BBC"
1030 DATA "SPECTRUM","GEOFF","ELEPHANT","XENOPHOBE",
     "WRINKLE","GRENADE"
```

A Case In Point

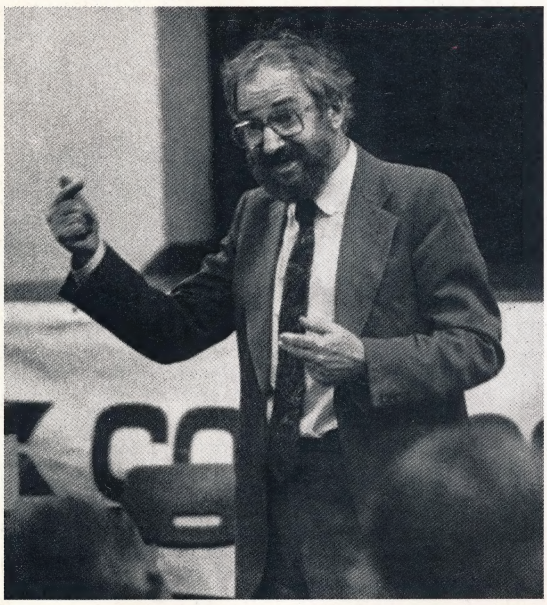
Notice that you must prompt the Hangman player to use either upper- or lower-case letters, according to the case of the mystery word

ADVANCE TO LOGO

We begin a series of articles about LOGO, a programming language designed primarily with education in mind. We look at the history of its development, the basic structural philosophy behind the language and the types of users it may appeal to.

Having examined BASIC and machine code programming in detail, we begin our course on other popular computer languages with a series of articles about LOGO. You may wonder why we have chosen this language as a subject for an extended learning programme. After all, there are many other languages that perform certain functions extremely well, and home computers are rarely supplied with LOGO. Nevertheless, LOGO does offer some very attractive features to the home computer user.

First of all, LOGO is one the best introductory languages available on any computer today. Of course, if you have been programming in BASIC, you may feel little need to know about an introductory language. Yet, even for the experienced BASIC programmer, LOGO can serve as an excellent introduction to 'structured' programming and to the use of procedures instead of statements. Secondly, LOGO is available on cartridge or cassette for most home computers. In fact, one of the best versions of LOGO available is written for the Spectrum and distributed by Sinclair. Finally, LOGO is a very powerful learning system. Though not an easy language to master by any means, LOGO is one of the few programming languages that is easy to start working with.



Founding Father
Seymour Papert, the founding father of LOGO, is shown here at a conference sponsored by Commodore in 1983. Papert is now associated with LOGO Computer Systems, Inc, (LCSI), which provides LOGO programs for the Sinclair Spectrum, Atari computers, and others

Meet LOGO

LOGO has two fundamental characteristics that make it such a powerful educational language. The first is that it is interactive: when you type in a command, you immediately see the results on the screen. This means it is easy to make progress (particularly for children and beginners) because you can check yourself at every step.

The second vital feature is that LOGO is extendable: complete operations are handled by lists of elemental LOGO instructions. These lists are called procedures. Once a procedure has been defined as a set of particular instructions, the name of that procedure takes on the status of a new LOGO command. From then on, the entire procedure can be executed simply by typing in its name. In this way, you can actually create your own commands in addition to those 'primitives' that are an inherent part of the language.

In programming with LOGO, most people tend to be more exploratory than with other languages. Sometimes they will take a fairly strict approach and define a specific outline from the very beginning. Sometimes they will start with a core problem and write a procedure to solve it, and then build a program around that procedure. It is possible to take a flexible approach to LOGO programming because there are usually several ways to arrive at a particular result.

LOGO's origins lie in the artificial intelligence language LISP, which was invented in the early 1960s to make it easier for computers to deal with complex data structures. Its name derives from the fact that it is a 'list processing' language, which means that its basic data structure is a list, rather than a character string or numeric array, as in BASIC. LISP's essential functions manipulate the data within a list. List elements can be simple symbols or whole lists. The advantage of this approach is that non-numeric data (such as a sentence) is more easily processed in this manner.

LISP relies heavily on the principle of recursion, whereby something (usually a function or procedure) is defined in terms of itself. In the case of LISP, the item being defined is always a list. These are not accidental characteristics of LISP, but arise from its origins in computer-based investigations of natural language and human intelligence. However, the language is not an easy one to learn, and in 1968 a group of people associated with the Massachusetts Institute of Technology (MIT) set about devising a language for children based on LISP.

The charismatic leader of the group at MIT was Seymour Papert. He had previously spent a number of years studying cognitive (learning)

development among young children with Jean Piaget (1896–1980), the leading educational psychologist of his generation. On moving to MIT, Papert began to work closely with an artificial intelligence expert, Marvin Minsky. In his work with LOGO, Papert attempted to bring the ideas of both his colleagues together, uniting theories of cognitive learning and artificial intelligence.

Work on LOGO continued throughout the 1970s, and other groups were set up to experiment with the new language. The most notable of these was based in Edinburgh. All of this development work was carried out in university research departments using mainframes or minicomputers. It was only with the arrival of microcomputers that LOGO became more widely available.

LOGO is a sophisticated language that needs a lot of memory, both for the code and as working space. The LOGO interpreters found on micros typically require around 30 Kbytes of memory, and another eight Kbytes or more for the graphics display. All of this before you even start programming! So, although it was possible to implement simple BASIC interpreters on home micros from the moment they were marketed, it was not until home computers with over 48 Kbytes of RAM were widely available that LOGO on micros became a feasible proposition.

But it was Seymour Papert's *Mindstorms* (Basic Books, 1980) that took LOGO out of the research departments and brought it to the attention of a much larger group of people. In his book, Papert develops a vision of how computers might be used in education. This is a result of the synthesis of three sets of ideas: theories of cognitive development, artificial intelligence and the movement in education towards child-centred learning. Papert wants to see children programming computers, rather than computers programming children (which, he argues, happens in most 'computer aided instruction').

The book looks forward to the emergence of a new 'computer culture', in which 'formal' ideas previously considered beyond the capabilities of children will be easily handled by them. They will be able to do this because of the way they have used computers to explore formal ideas. It is this active, co-operative (pupil-to-pupil and pupil-to-teacher) and unstructured exploration of ideas that constitutes the 'LOGO philosophy' underlying the language's use in education.

Papert writes and convinces by the sheer power of his rhetoric. However, there are a number of serious problems with the theory. There is very little experimental evidence to back it up, despite a number of studies; Piaget's theories of cognitive development are turned into a prescription for education in a way that Piaget never intended; and there are problem-solving areas (even in mathematics!) that LOGO doesn't cover.

As teachers use LOGO more widely in the classroom, they are finding that not everything works in the way that Papert describes, and they aren't getting the results they had hoped for. There is a danger of disillusionment, but once we set aside the over-enthusiastic claims of what the language can do, LOGO still remains an excellent way of introducing computer concepts, of exploring certain kinds of ideas and developing problem-solving skills.

Authorised Versions
Shown here are the most comprehensive and well-documented versions of LOGO for the Commodore 64 (Terrapin-MIT), Sinclair Spectrum, and Atari computers (LCSI). Although expensive, these are the manufacturers' authorised versions of LOGO and will most closely resemble the original MIT language. Commodore 64 LOGO is available on disk for £34.95; Atari LOGO comes on cartridge for £59.95; and Sinclair LOGO is a cassette-based version for £39.95. There are less expensive LOGO programs on the market. Some, particularly for the Spectrum, cost as little as £8.95

LOGO on present day micros has too little workspace and runs too slowly. To some extent it is a language that is waiting for the hardware to catch up with it. But as a language for learning it has no serious rivals.

Who Is LOGO For?

Who can benefit from learning to program in LOGO? We feel that many people, even experienced programmers, can learn a great deal from LOGO programming, including:

- Anyone who is new to computing, or new to programming;
- Anyone who likes playing with computers, and thinks computers ought to be fun to use;
- Anyone frustrated by a lack of expressive power in another programming language;
- Anyone who has an interest in thinking about thinking, learning, or teaching;
- Anyone who wants an insight into more advanced areas of computing, especially those that are related to the study of artificial intelligence.

Having said this, we must remind you that, like BASIC and machine code programming, LOGO is not for everyone. Specifically, LOGO might not be the best language for:

- Anyone who thinks using computers is 'work'. Some languages are designed for work, as are cart horses. But a cart horse is hardly the one you would choose for an afternoon ride in the countryside;
- Anyone who needs or expects a great deal of speed from the computer as it processes instructions. LOGO uses a lot of memory, and runs slowly on the present generation of microcomputers. (On comparable programs, LOGO can run at half the speed of BASIC.)

Even for these groups, however, a knowledge of LOGO can be very valuable. LOGO can be used to sketch out a solution to a problem and prepare it for translation into another language.

Coming soon . . .

the Turtle

EAPROM

The *electrically alterable programmable read-only memory* is one of the newest types of semiconductor memory. To understand its operation, we must compare the EAPROM with other forms of memory. A normal ROM is mask-programmed, meaning that its contents are fixed during the manufacturing process. A PROM (programmable ROM) is like a blank ROM; a 'PROM burner' allows its contents to be programmed by the user, but once programmed these contents may never be changed. An EPROM (erasable PROM) has a quartz window on the chip's surface. Exposing this window to ultra-violet (UV) light will erase the contents and allow the user to re-program the device. EPROMs are used widely in development systems and may be used instead of ROMs, which are economic only when manufactured in thousands of units.

The EEPROM (electrically erasable PROM) is similar in operation to the EPROM, but uses electricity instead of UV light to erase the chip's contents. The EAPROM, however, allows individual memory locations to be altered, although writing information to the chip is considerably more complex, and hence slower, than reading information from it. EAPROMs are therefore used only in applications in which the memory contents change little and where these contents must be retained when the power is switched off.

EDGE CONNECTOR

The *edge connector* is the simplest form of interface connection, and is a favourite among manufacturers of less expensive home computers. The contacts are printed onto the edge of the printed circuit board by using the normal etching process, and the PCB is then trimmed to allow the contacts to protrude by about one centimetre. Well-made edge connectors use gold-plated contacts, as copper oxidises when exposed to air and this can result in bad connections. More expensive machines dispense with edge connectors and use purpose-designed interface sockets such as DIN sockets or 25-way D-connectors.

EDITOR

An *editor* is a program that allows the user to create and alter patterns of symbols. These symbols may be graphics, program listings or, in the case of a word processing editor, English text. Electronic editing relies on several basic functions: insertion and deletion of text, overwriting, etc. More sophisticated editors allow the same function to be performed in a variety of ways — the user may delete a character, a word, a sentence, a paragraph or a page, for example — and some incorporate a 'search and replace' function that allows any symbol, word or phrase to be replaced by another. A 'full-screen editor' permits alterations to be made at any position on the screen. The more common line-based editor

Copy Editor
On the Spectrum, a copy of the line indicated by the cursor is edited at the bottom of the screen

requires the user to specify the line to be altered, and permits editing only on that line.

ELECTRONIC MAIL

Communication is rapidly becoming the most important factor in business microcomputing — today's users rely increasingly on information that may be retrieved via the telephone network. One of the most popular off-the-shelf applications is *electronic mail,* which is a facility for sending documents and messages from one micro user to another. This requires that each user has a microcomputer (preferably with printer attached), a modem and an electronic mail software package. Short messages may be typed on the keyboard while the computer is on-line. To save on telephone charges, however, longer documents are usually prepared off-line by using a conventional word processing package. The communications utility then reads the finished file from disk or cassette and transmits it character by character.

ELECTROSENSITIVE PRINTER

Popularly known as 'thermal' printers, *electrosensitive printers* are cheap to construct, quiet and relatively fast in operation. The major drawback with these devices is the special paper that is required; this is silver-coated and totally unsuitable for serious use as it is often difficult to decipher text printed on it. It is also considerably more expensive than plain paper. As a result of these limitations, electrosensitive printers are most often used as cheap devices for listing programs — a good example is the Sinclair ZX printer, which uses narrow rolls of paper giving only 32 characters per line.

Printed characters are built up from dots, but the moving pins of a conventional dot matrix printer are replaced by a solid-state array of needles, to which an electrical charge of several thousand volts is applied. The resulting spark burns off the metallic coating on the paper to leave a small black dot. The high voltage is not a problem as the system does not produce a large current and hence is not dangerous to the user.

NEW MUSICAL EXPRESSIONS

In the introduction to this series, we saw how the use of electronics in music-making has developed. To begin with, simple tones were produced with oscillators, and the range of available sounds was limited. Today digital encoding of sound — called sampling — is available, and musicians can use any sound to produce music.

Despite the many developments in this field over the past 60 years, it is worth remembering the simple controlled voltage characteristics of an oscillator. When any physical object — whether a bee's wing or a human vocal cord — vibrates, the surrounding air expands and contracts very rapidly, producing a waveform that is interpreted by the human ear and brain as sound. If an electrical voltage is applied through a modulator (like a car's induction coil) to a tiny strip of metal, the metal will vibrate, creating the simplest wave form — the sine wave. The pitch, or frequency of vibration, of the resulting oscillation depends on the voltage applied and, to a lesser extent, on the density of the metal strip. This tiny sound-generating unit is called an oscillator. Voltage control has been the primary method of producing synthesised music for decades.

The MIDI interface, first announced in 1983, is a unit that is designed to allow one digital system (such as a computer) to control another — a synthesiser, for example. Its development is a result of the advances in electronic music-making over the last decade or so.

For several years, recording studios have contained many different pieces of sound-processing equipment — an impressive array of filter and reverberation units is often seen as proof of a studio's worth. Similarly, in live performance, a synthesiser player in the 1970s had to be entirely surrounded by banks of keyboards, each with a multitude of controls.

When considering what happens in a well-equipped recording studio, it is useful to think of the party game 'Chinese Whispers'. In this game, a sentence is passed from person to person. The last player then recites what he has heard of the original sentence. A straightforward sentence may have been changed into a collection of nonsense words, or vice versa. A similar process occurs in the recording studio, but here the original sentence is a collection of musical sounds. And instead of a chain of listeners, each producing a garbled version of the original, there is a group of sound-processing units, each one controllable and doing a specified task. Anyone using this equipment would probably use a central plugboard to make connections, or else connect units together directly. The controls on each unit would be calibrated manually in a matter of seconds by a skilled sound engineer, but the problems of synchronisation and communication inherent in such connections are easily imagined.

Twin Set
Alannah Currey, Tom Bailey, and Joe Leeway, of the Thompson Twins (shown seated front to back). Originally a seven-piece 'arty' band, they now perform as a trio with taped, sequenced rhythms as backup

The keyboard player of the 1970s had a different problem. Here, the immediate difficulty was not how to produce a sound from each instrument in succession — the player had only to move a hand from one keyboard to another. It was more likely that he would experience difficulties when two keyboards were to be played at once, but harpsichordists and church organists had known how to do this for centuries. Even complex music like a Bach fugue could be performed on one keyboard alone, and, anyway, most studio work would not have been done in one 'take'. Instead, each synthesiser part would be recorded on its own, with subsequent parts superimposed on the first, using different parts of a multi-track tape. Performing such music however, needed one skilful artiste, or two average musicians, or tape loops and a sound engineer.

But the most important development of the 1970s was in the sound-generating units *inside* the

Control Centre
The Roland MP401 MIDI interface is a sophisticated unit that connects microcomputers to digital synthesisers. It controls all external synchronisation functions, internal and external timing, tape input/output, and synthesiser output. This means the host computer is responsible only for the sending and receiving of commands and memory management. The MP401 costs £160, and works with the IBM PC and the Apple II and IIe. According to Roland, a Commodore 64 version will be available in the near future

synthesiser, and in instrumental techniques in general. A good example of this is the emergence of the keyboard-controlled bass synthesiser. In the 1960s, the Motown style of pop/soul music relied heavily on the electric bass. During the 1970s, funk bass players developed a level of virtuosity that rivalled that of lead guitarists; by the end of the decade this style could go no further and the keyboard-controlled bass synthesiser emerged. This led to new problems for the keyboard player, who was now required to take on the functions of the bass-player — working with the drummer to 'anchor' the rhythm section in strict time — and who was now not playing 'keyboard' music. As new analogue synthesisers appeared on the market, trumpet, saxophone and drum sounds became available. More and more keyboard players turned to a simple device to deal with these new responsibilities. This was the sequencer.

A sequencer is a device that co-ordinates several independent strands of music into a unified work according to a specified pattern. It operates by using controlled voltages that are fed through an oscillator to produce a series of tones of different frequencies. The higher the voltage, the faster the metal strip vibrates, and the resulting waveform is heard as a high-pitched sound. A sequencing unit is used to control the oscillator. This is necessary because music is rarely composed of nonstop sound sequences — short gaps or long silences are required, both to create rhythmic patterns and to form the overall structure of music. A gap in a sequence pattern is caused when the control unit sends a zero voltage to the oscillator. The purpose of 'sequencing' is to ensure that the gap occurs in exactly the same place each time the pattern repeats.

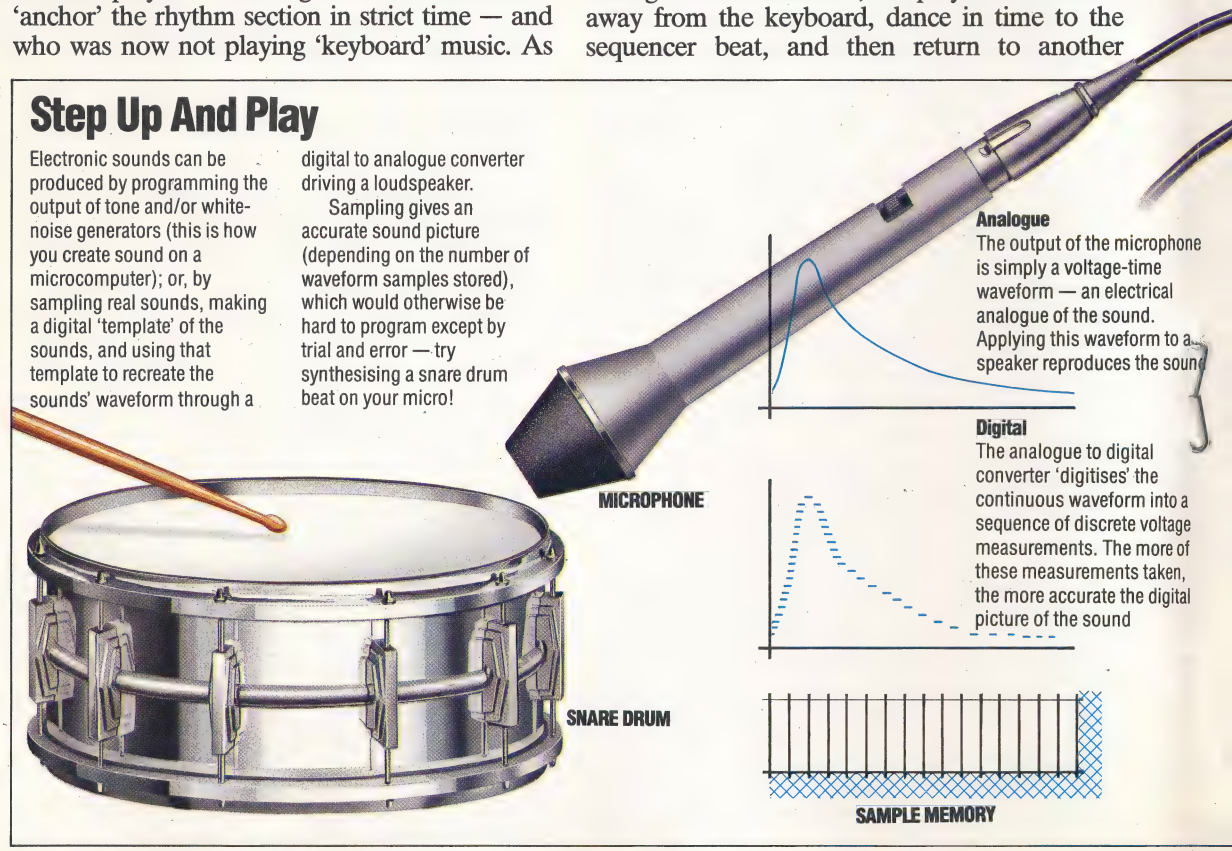
Few synthesisers in the late 1970s had complete sequencing facilities, but musicians were quick to use what was available. The pulsating disco music produced by Giorgio Moroder with Donna Summer is very much sequencer music, and the new British synthesiser bands developed a completely new style to match the new equipment. No longer was it necessary to play every single note with a series of grand gestures, in the style of Rick Wakeman. Instead, a whole sequence, or riff, could be started or stopped by changing one setting. In the meantime, the player could move away from the keyboard, dance in time to the sequencer beat, and then return to another

STEVE CROSS

Step Up And Play

Electronic sounds can be produced by programming the output of tone and/or white-noise generators (this is how you create sound on a microcomputer); or, by sampling real sounds, making a digital 'template' of the sounds, and using that template to recreate the sounds' waveform through a digital to analogue converter driving a loudspeaker.

Sampling gives an accurate sound picture (depending on the number of waveform samples stored), which would otherwise be hard to program except by trial and error — try synthesising a snare drum beat on your micro!

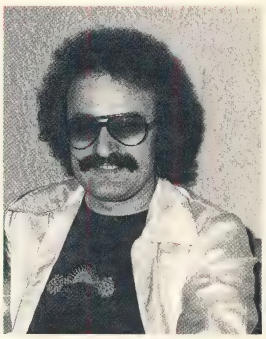

MICROPHONE
SNARE DRUM
Analogue
The output of the microphone is simply a voltage-time waveform — an electrical analogue of the sound. Applying this waveform to a speaker reproduces the sound
Digital
The analogue to digital converter 'digitises' the continuous waveform into a sequence of discrete voltage measurements. The more of these measurements taken, the more accurate the digital picture of the sound
SAMPLE MEMORY

keyboard to play a melody or group of chords. In the mid-1980s, the whole performance style of a group like the Thompson Twins is influenced by the existence of the sequencer.

When digital synthesisers first appeared, their design was often modelled on their analogue predecessors. Musicians found that their sequencing skills were further developed by the new instruments, and it is in this area of digital control that most interest has been generated. This is demonstrated by the recent popularity of the Linn drum machine, one of the first units to use sampled sound — in this case provided by top American session drummer Steve Gadd. On the Linn machine, drum patterns are recorded in digital form in the same way that computer data is recorded on floppy disks. The resulting sequences of ones and noughts are then encoded onto ROM chips. By accessing a particular chip, a musician or producer can reproduce the original sound as if it were being played live. The great benefit of digitising the sound is that output may be altered at the console, diverging from the original pattern in time, rhythm, volume, etc.

By now our two original examples — the recording studio and the onstage synthesiser player — have a number of features in common. In addition to recording music, studio work reflects the video boom that has occurred in the last few years. Video has brought a new style and consciousness to image-making: producers demand that the accompanying music reflects this.

If the music accompanying a video is produced by conventional instruments, synchronisation with the on-screen image is very similar to techniques used with film. If, however, the music is made up of individual sounds and sequences produced by digital synthesisers, there is much that can go wrong. Let us imagine that on a particular video a vase of flowers is dropped to the ground, where it smashes. The musician working on the score has produced a sequence that accelerates up to the point where the vase drops, and given a percussive chord for the moment the vase hits the ground. Recording starts and the sequence begins, but it soon becomes clear that the rate of acceleration has been miscalculated and the sequence ends while the vase is still on the table. The musician then tries the percussive chord, which is recorded on a different tape track. Recording starts again, and this time the chord is recorded a split-second too late. The musicians need a way of linking up the digital instruments so that everything happens at the right time. What is needed, then, is a digital interface.



COURTESY OF RECORD MIRROR

Dynamic Duo
Donna Summer, with producer Giorgio Moroder, was one of the first pop artists to use electronically produced rhythms and synthesisers in recorded music

The synthesiser player has similar problems, this time in a live performance. His equipment includes two digital synthesisers, made by different manufacturers, and a Linn drum machine. He has sequences set up on one synthesiser and on the Linn, but as they do not keep in time together he usually ends up running the Linn automatically and playing the other manually. The result is that his second synthesiser, bought for the quality of its pre-set sounds, remains untouched. This musician needs a way of linking up his instruments so that all the sequenced material occurs in the right place. He also requires that the sequencer on his first synthesiser should play the pre-set sounds on the second. Furthermore, whatever equipment he uses should be applicable to more than just his own synthesisers — he may well find himself in the studio mentioned above that has so much trouble with video synchronisation!

In the next instalment of this series, we will look in detail at the MIDI interface and examine other examples of sequencing techniques.

Electronic Drum Beat
A drum phrase can now be constructed in any order on the synthesiser keys (each corresponding to a beat in the bar). On playback, a scan sweeps the keyboard at a pre-set time signature (or step time), playing a digitised drumbeat if that key has been hit, and illuminating its LED. The phrase can thus be created and edited layer by layer.

THE HUMAN FACTOR

An important aspect of program design is the 'man-machine interface' — the part of the program that deals with the transfer of information from user to program and vice versa. Here, we investigate the factors to be considered when designing this interface.

Computer programming was for many years a mysterious topic that was understood only by professionals who were prepared to devote much time and effort to the subject. Before the advent of the microcomputer with its typewriter-style keyboard, programs were often entered one byte at a time via switches on the computer's front panel, or by punching holes in tapes on a teletype console.

Today's user is, by contrast, a pampered creature. Manufacturers no longer expect the computer owner to struggle with machine code, and the phrase 'user-friendly' was coined to indicate that micros may be used and programmed by anyone, regardless of experience. In 1982, the Alvey Committee, in a report entitled *A Programme for Advanced Information Technology*, identified the man-machine interface (MMI) as one of the four main areas of research and development, together with software engineering, very large scale integrated circuit (VLSI) design and knowledge-based systems.

In any application, the interaction between computer and user, where data or instructions are passed between the two, is of paramount importance. This 'dialogue' is conducted through the computer's input/output (I/O) devices, with the keyboard serving as the main source of input and the display screen providing the output. Joysticks, paddles, mice, touch screens and other devices may also be used for input, while the computer can utilise a printer, sound (or speech) generator or even a robot to express the output.

In addition to any constraints imposed by the I/O devices used, the dialogue between user and machine is influenced by software. For example, the computer's operating system (OS) controls many details of the screen and keyboard operation. The rate at which keys repeat when held down, and the delay between repetitions, is set by the operating system, which also buffers keystrokes to allow the computer to store characters that have been entered faster than they can be displayed. This is very important as it affects the speed at which the user may enter information into the computer. The buffer size is critical and should be known by the user — the CP/M operating system, for example, buffers a single keystroke; many home machines buffer 10 strokes or more.

But keystroke buffers may cause problems. An experienced user who is working with a menu-driven system may know in advance that the menu choices he requires are 2 from the main menu, 5 from the next menu, then 3, 4, 6, etc. Because he is familiar with the system, he types his choices at great speed. With a 10-character buffer, the user will end up where he wanted to go because the keystrokes will all be 'remembered' in the correct sequence. With a one-character buffer, the time taken to display the second menu may be longer than the time taken to type the sequence. Thus, instead of selecting choice number 5 from this menu, then 3 and so on, choice number 6 alone is made (because this is the only character held in the buffer) and the system stops there.

But a large buffer can also lead to problems. A menu program that takes a long time to react to a keypress (this may occur if the choice leads to a file being read) may cause the user to think that nothing is happening. The natural response is to try the last choice entered, then press an assortment of keys until there is a response. This

CP/M

may lead to the program attempting to process the spurious characters held in the buffer; the results may be surprising!

'Garbage collection', which involves clearing the computer's memory registers to free working space is another source of problems. This can make a program appear to 'hang' for long periods, during which the user may again try to take corrective action. Garbage collection is likely to cause problems in large programs that do a lot of string handling. Some versions of BASIC allow the

programmer to force a garbage collection; it is a good idea to do this at frequent intervals — but the user should be given a 'please wait' message as the computer will appear to be doing nothing while garbage collection is being dealt with.

The way that a programming language handles input and output will influence the design of the interface between computer and user. The superior string-handling facilities of BASIC will lead to more sophisticated use of strings in the human-computer dialogue than is allowed by languages like PASCAL. BASICS that have built-in commands for cursor addressing will encourage better screen layouts than those that do not. The same holds true for BASICS with graphics commands. BASIC is well supplied with input/output commands — INPUT and PRINT are fine for simple programs. But for real control of input (the kind needed to produce a form containing protected input fields, for example) try experimenting with GET$, INKEY$, INPUT$() and similar commands. PRINT USING is an extremely versatile command for formatting output; it is invaluable for aligning decimal points and for justifying columns of text.

Micronet 800 'Log-on'

USER PSYCHOLOGY

The user is the most unpredictable element in any man-machine system. Like any other component, though, the user has certain performance characteristics that must be understood before the interface is designed.

People share with computers the basic characteristic of being 'information processors'. However, human beings have inherent limitations on the amount of new information they can hold in 'working memory' — it has been reckoned that for most types of information around seven different items may be held in the brain at one time. The size of these items depends on how meaningful or well structured they are. If the information to be remembered consists of random characters, each item will consist of no more than a single character. But if the characters are not random but form common surnames, each item remembered could be an entire name. Increasing the structure of the information in this way increases the user's ability to remember and make use of it.

There are several ways of helping people to structure information when using computers. One method is to relate data to familiar, well-understood structures — this is the way that the Lisa-style 'desk-top' metaphor works. Similarly, a financial spreadsheet package may be organised to look like a book with pages, indexes, etc. Another method is to train the user to understand unfamiliar structures. By repeatedly showing examples, and explaining topics in depth, the program itself may be used to teach the user how the information should be structured. The

Apple Macintosh

IAN McKINNELL

drawback with training of this type is that it is expensive in both time and effort. Detailed instructions, 'help' screens and 'signposts' may provide a type of on-line training, but these can be difficult to use efficiently.

Finally, presenting information in recognisable patterns can help the user to understand the program. This can be done by using colour or layout to lead the eye to the desired information. To understand what this means, consider colour-coding as it is used in Prestel and similar videotext programs. On a typical page, the heading and 'footer' will be set in blocks of the same colour; there will be a single background colour, and text will be displayed in two other colours, with alternate paragraphs in each colour. Key words may be highlighted by using yet another colour. The purpose of this is to allow the user to select only the information required and to ignore whole sections of the page if these sections contain information of no immediate value. Colour-coding can be confusing if it is over-used, however, and tests have shown that people may waste time reading and re-reading paragraphs to try to understand the significance of an entirely arbitrary colour change! A good rule of thumb is: never use more than four colours at once.

Three Degrees
These photographs of microcomputer operating systems illustrate three varying degrees of user-friendliness. In the first photograph, a new user is attempting to communicate with the CP/M operating system. CP/M has no built-in 'help' features, so requires a thorough knowledge of commands before it can be used properly. Our second example is a menu-driven system — the 'Log-on' menu for Micronet 800 on the BBC Micro. Options are clearly numbered, and the user makes his selection by entering the appropriate number from the menu. The screen does not offer a great deal of information, so the user must understand the options before he can make use of them. Our final photograph shows the Apple Macintosh operating system, which provides visual clues and graphic displays, as well as simple, easily understood menus

HOT WIRED

A microcomputer's user port is the gateway through which we can monitor and control aspects of the world beyond the confines of the machine. In this introduction to a new Workshop series, we will look at how to access your user port so that you will be able to monitor physical phenomena such as heat, light and force, and control external devices — including a simple robot.

Many popular home micros have user ports that allow entry to the computer's memory map by way of a series of electrical connections. The basis of all digital systems is that the binary system of ones and zeros can be easily represented by two voltage levels. Normally, a zero is represented by 0 volts and a one by +5 volts.

Each memory location is made up of a group of eight individual cells, each cell having a voltage level of 0 or 5 volts. The pattern of these voltage levels thus determines the number that is stored in that memory location. If any cell has a voltage level of +5 volts then we say that the cell is *set high* and if the cell has a level of 0 volts then the cell is said to be *set low*. The external connections of the user port are electrically linked to one or more locations in the micro's memory, and by reading values from, or writing values to, these locations we can monitor or control electrical systems outside the computer.

There are two types of user port connections. Some ports have separate groups of pins (eight for input and eight for output) linked to two locations in memory. Others use the same eight pins for both input and output. In this part of the series, we will look at the second type of configuration, as used by the BBC and Commodore 64 micros.

DATA DIRECTION REGISTERS

In addition to having a location linked to the eight pins of the user port, micros with bi-directional ports make use of a second memory location, known as the *data direction register* (DDR). This register determines whether each of the eight lines is to send or receive data. A one in the DDR sets a

Plugging In

We begin our project by making leads for the BBC Micro and Commodore 64. These will be used to connect the machines to the outside world via a common format of eight data lines with an earth line on either side. You will need:

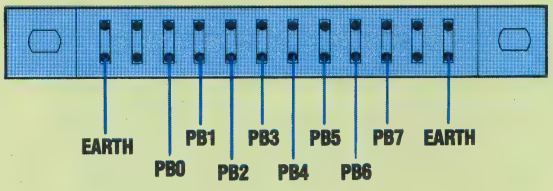
BBC Micro
- 20-way IDC plug
- 20-way IDC ribbon cable (about 1 metre)
- Soldering iron and solder

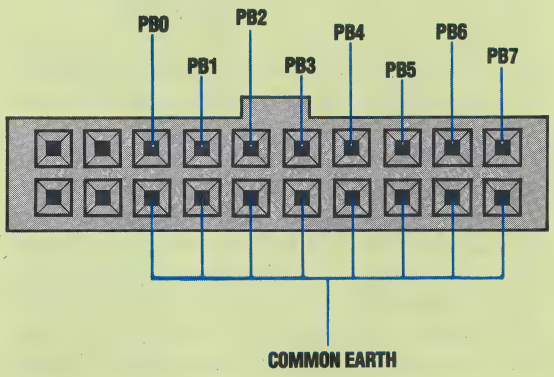
Commodore 64
- 24-way 0.15″ edge connector
- 10-way ribbon cable (about 1 metre)
- Soldering iron and solder

The machine manuals' port pin-out diagrams show the location of the 10 lines (two earth, eight data) that we need. The BBC Micro's IDC (insulation displacing connector) plug has a locating lug on one side, and splits into two unequal pieces: hold the plug vertically with the lug away from you and its clamp section uppermost, and thread about an inch of ribbon cable through it with the red stripe to your right. Close the plug onto the cable with steady pressure (perhaps from a vice or clamp). Separate and trim 10 lines at the other end as shown, then strip and tin the ends of the remaining lines (see page 44).

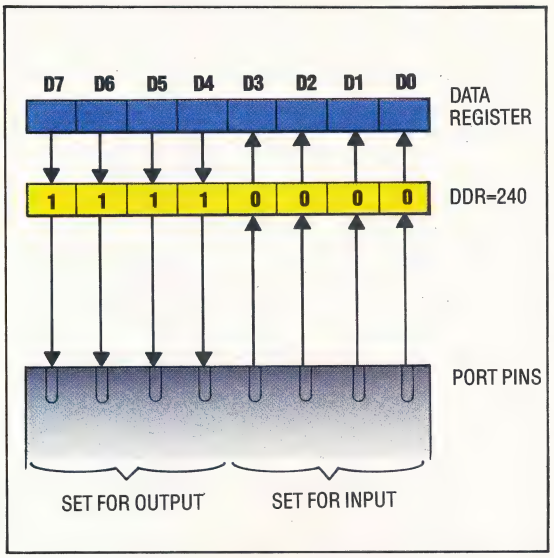
For the Commodore 64, mark one side of the plug clearly as the top (and always have this side uppermost on connection to the machine). Strip and tin both ends of the 10 lines, and solder the cable to the bottom pins of the edge connector in accordance with the pin-out.

Use the test programs and a multimeter (see page 87) to check your leads





line to output mode, and a zero allows input to be received. To set all eight user port lines to output, the DDR would have to be set to 255 (i.e. 11111111 in binary). Similarly, all eight lines can be set to accept input by setting the value in the DDR to zero. The eight lines can be configured in any combination of input or output lines by setting the appropriate value of the DDR. For example, the most significant four lines of the user port could be set to output mode, and the least significant four to input by placing the value 240 (i.e. 11110000 in binary) in the DDR.



The data and data direction registers have the following addresses:

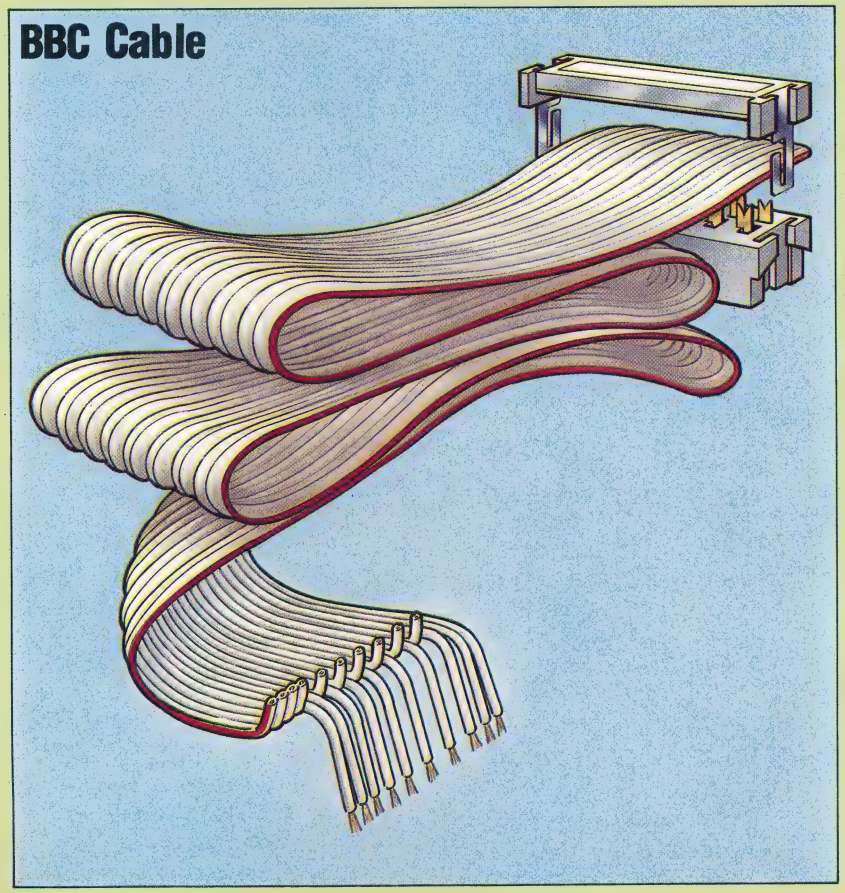
Micro Type	Data Register	Data Direction Register
BBC Micro	&FE60 (65120 dec)	&FE62 (65122 dec)
Commodore 64	$DD01 (56577 dec)	$DD03 (56579 dec)

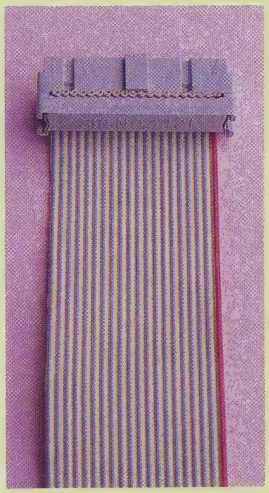
The following program sets the user port so that all eight lines may be used for input, and displays the data register contents:

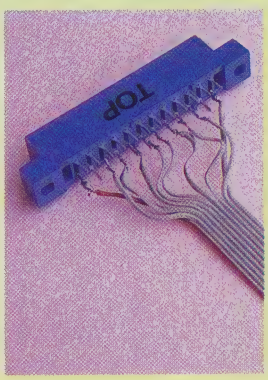
```
4 REM**********C64***********
5 REM*     DATREG DISPLAY      *
6 REM**********C64***********
10 DIM A$(10):A$(0)="E":A$(1)="H"
20 DATREG=56577:DDR=56579
30 POKE DDR,0: REM = INPUT ONLY
50 :
100 PE=PEEK(DATREG):GOSUB 500
150 PRINT"DATREG =";PE;"=";B$
200 GOTO 100
300 :
499 REM************************
500 REM*  BINARY CONVERT S/R *
501 REM************************
550 B$="":N=PE
600 FOR D=1 TO 8
650 N1=INT(N/2):R=N-2*N1
700 B$=A$(R)+B$:N=N1
750 NEXT D:RETURN
```

On the BBC Micro, make these changes:

```
20 DATREG=&FE60:DDR=&FE62
30 ?DDR=0
100 PE=?(PE):GOSUB 500
```



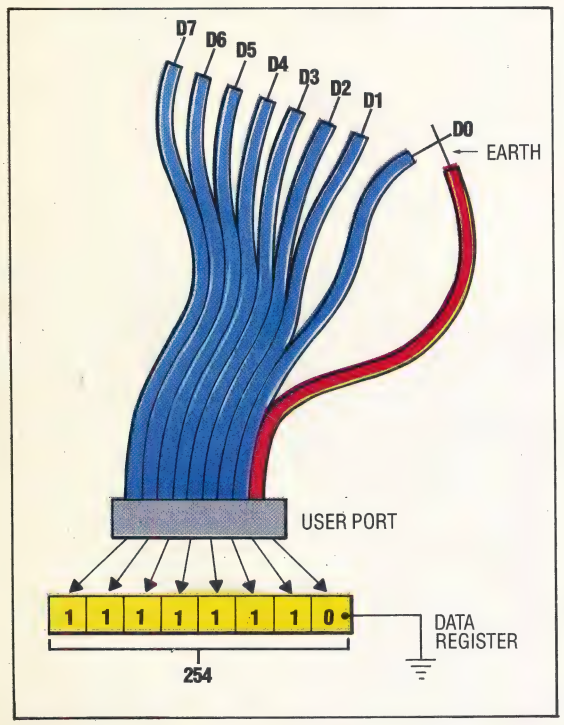




Leading Lines
The BBC Micro and Commodore 64 user port leads

RUNning the program shows the normal contents of the data register to be 255 (shown on screen as HHHHHHHH), which means that all eight lines are high, and all eight cells of DATREG are at voltage level +5. If you now plug a lead into the user port, you can use it to change the voltage levels on the lines, and, therefore, change the numerical contents of DATREG.

We have wired 10 lines to the user port: eight of these are data lines — one line for each bit of DATREG — and the others provide a 'case ground' or 'earth' for the system. Touching a data line to an earth line will pull the data line's voltage level down to zero, thus changing the voltage levels in DATREG. If you earth D0, for example, while the program is running, you will see that DATREG now contains 254 (shown on the screen as HHHHHHHE), meaning that the least significant line is at *earth* voltage (0v), while all the others are *high* (+5v). You might like to try different combinations of lines to assure yourself that you can, by these means, make DATREG contain any number between zero and 255.



WRITING SOFTWARE

The essence of computer decision-making is the testing of numeric values or conditions and branching according to the result obtained in the test. Therefore, it is a simple matter to design software that can monitor physical activity via the user port. By testing the value of the data register and taking the appropriate action, the computer can respond to external changes. A simple example would be to design a program that checks to see if a telephone number has been 'dialled' correctly. Dialling can be done by grounding out certain data lines connected to the user port to produce the correct numeric value in the data register for each digit in the telephone number.

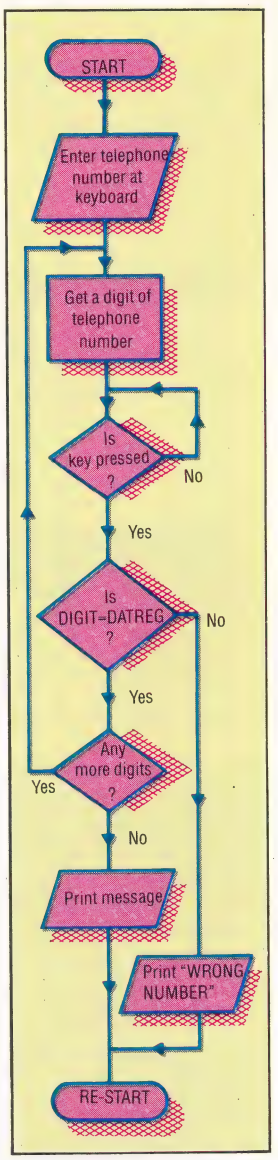
Because of the difficulties involved in simultaneously grounding a number of data lines, the program waits for the user to press a key on the keyboard before analysing the contents of the data register. The logic for the program is shown in the accompanying flowchart and is quite easy to follow.

```
100 REM**********C64***************
101 REM*   TELEPHONE NUMBERS      *
102 REM**********C64***************
120 :
130 DATREG=56577:DDR=56579
140 POKE DDR,0:REM = INPUT ONLY
150 :
160 INPUT"TELEPHONE NUMBER";TN$
170 :
180 FOR K=1 TO LEN(TN$)
190 DG$=MID$(TN$,K,1): REM GET DIGIT
200 IF DG$<>" " THEN GOSUB 500
210 NEXT K
250 :
300 PRINT"-------SORRY-------"
310 PRINT" NO ONE IS IN AT ";TN$
320 PRINT" PLEASE CALL LATER"
350 PRINT:PRINT:RUN
400 :
500 REM**CONVERT & CHECK S/R**
550 DG=VAL(DG$)
600 PRINT"SET UP DIGIT ON THE LINES"
650 PRINT"     AND HIT ANY KEY"
700 GET GT$:IF GT$="" THEN 700
750 PE=PEEK(DATREG):IF PE=DG THEN
PRINT DG" OK":RETURN
800 :
850 PRINT"???WRONG NUMBER???"
900 PRINT TN$"<>"LEFT$(TN$,K-1):PE
950 PRINT"??? TRY AGAIN  ???"
999 PRINT:PRINT:RUN
```

On the BBC Micro, make these changes:

```
130 DATREG=&FE60:DDR=&FE62
140 ?DDR=0
700 A=GET
750 PE=?(DATREG):IF PE=DG THEN
PRINT DG" OK":RETURN
```

In this introductory article, we have looked at inputting information from an external source in order to affect the flow of control within programs. In the next instalment, we shall look at output via the user port and the design of a simple computer control system.



Crossed Lines
The Telephone Numbers program accepts as input a telephone number (any format, any length), and checks its digits one-by-one against the contents of the user port memory location. Spaces in the input number are ignored. The program waits for a keypress before comparing an input digit with the user port digit

Program Exercises

Using our telephone number program as an example:

1) Write a program to simulate the action of a burglar alarm.
2) Write a program to count the number of pulses received by the user port within a given time period.
3) Modify your last answer to keep a separate count for each of the eight user port data lines.
4) Write a program to simulate the action of a combination lock.
5) Write a program to change the colour of the computer's screen from the user port.

BLACK MAGIC

Synapse Software's Necromancer, written by Bill Williams, can be compared to a 'well-made play'. The scenario is divided into three acts: the seeds of the drama are literally sown in the first act, and the plot moves remorselessly to a climactic encounter between good and evil.

The Overture Both Atari and Commodore 64 versions are easily loaded: the slow haul of the Commodore's sluggish disk drive is made tolerable by the program producing a range of curious noises — rather like an electronic orchestra tuning up. The title graphics are accompanied by introductory music that uses the sound capabilities of each machine to the full.

Act One: The Forest The drama begins in 'the age of darkness' when 'Tetragorn, the evil wizard, reigns supreme'. You play the part of Illuminar, a druid described as 'the defender of truth and protector of the human race'. As you can imagine, this isn't an easy task. Play starts with the druid appearing in a dark open space. An aura of stars protects you from hundreds of little ogres that march relentlessly across the screen waving giant cutlasses to an insistent musical accompaniment. You use the joystick to control a magical 'wisp' that flies around the screen destroying the ogres and notching up points before returning to your hand.

Placing the wisp in the desired space and pressing the joystick button enables you to plant trees in an attempt to create a forest that will aid you later in the game. You must protect the seedlings from the ogres' cutlasses and Tetragorn's spiders, all the while keeping an eye on your strength, which is sapped by the sting, or invigorated by the death, of a spider. After five levels of play, the first act ends in the spiders' attack, which quickly exhausts the druid's strength. The program then freezes the action, counts the number of trees you have grown and places the druid in the next act.

Act Two: The Vaults It is here that the spiders hatch. There are five different levels, each one containing two layers of four vaults. Inside the vaults are spider eggs, which flash in different colours when they are about to hatch. Also on-screen is the 'tree bin' (containing the trees you grew in the first act). You must use your wisp to release and move a tree to a position above one of the vaults; if you're quick, the tree will grow roots and smash through the top of the vault to destroy the eggs before they hatch.

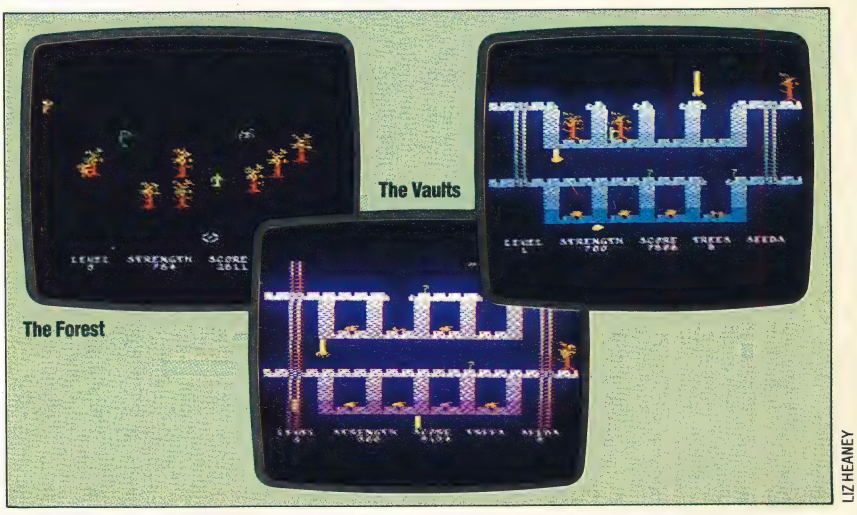
An added danger is supplied by the 'hands of fate'; these reach down from the ceiling and grab at anything underneath them — druid, trees or question marks. The latter are used to represent mystery prizes, and one must be acquired before you can lower a ladder to the next level.

Act Three: The Necromancer's Lair The climax of the drama is enacted in an eerily dark graveyard. The tombstones mark the many graves of the Necromancer and must be destroyed to prevent his re-incarnation. The wisp is powerful enough to kill each incarnation, and the gravestones themselves disappear if you move the druid on top of them. But the battle between good and evil isn't that simple, as all the spiders that escaped you in the second act are transformed into 'zombie' spiders and come to the Necromancer's defence.

It is imperative to reserve plenty of energy for the Lair, because here you must have all your wits about you. Certainly, the last act can be highly frustrating, as it is not possible to access it independently of the rest of the game.

Of the two versions of the game, the Atari implementation is the better. The Commodore version is slower and contains slightly less spectacular graphics and sound. Both versions can suffer from joysticks that give jerky and slow responses. These quibbles aside, however, owners of either version should find this a magnificent and enchanting — albeit expensive — game.

Necromancer: For Atari 400/800 (32K), £28.95, for Commodore 64, £28.95
Publishers: Synapse Software, 5327 Jacuzzi St, Suite 1, Richmond, CA 94804
Authors: Bill Williams (Atari), Scott and Steve Coleman (Commodore 64)
Joysticks: Standard Atari/Commodore joysticks
Format: Disk and cassette



LIZ HEANEY

Visual Treat
Necromancer combines sprite grahics with high-res background displays to produce dazzling screens at all three levels of skill. The glittering tree-tops, rapidly-darting druid's wisp, the movement of the spiders and ogres, and the necromancer's fire make Necromancer a stunning visual treat as well as an exciting game

HIGHLY PROCESSED CODE

Our machine code course continues with the first in a series of articles that takes a close look at the Assembly language of the 6809 processor, which is used in the Dragon and Tandy Color home computers. We begin with an explanation of the role of the registers in the functioning of the processor.

A microprocessor can be regarded as having three main components: the registers, which are parts of memory inside the processor; an ALU (arithmetic and logic unit) where certain simple mathematical operations can be performed on the data stored in the memory; and a control unit that makes everything happen in the right sequence at the right time.

At its lowest level of operation, the microprocessor responds to voltage signals applied at some of its external connections to change its internal state (the contents of its registers) or to send and receive other signals. By representing the presence of a voltage as a one and its absence by a zero, we can think of these signals — travelling back and forth between the processor and memory, or within the processor itself — as numbers in binary notation. In this way, we can program the processor by applying a sequence of numbers as instructions for it to act on. The very lowest level of programming, therefore, involves thinking in terms of binary (or hexadecimal) numbers. It requires a knowledge of the effect of each number, or instruction code, on the processor.

An eight-bit processor, like the 6809, can send and receive binary numbers with eight bits, which gives a range of decimal numbers between 0 and 255. Many of the numbers are used to refer to addresses of memory locations, which on most eight-bit processors are given as 16-bit numbers (allowing for a range of memory locations between 0 and 65535). Of course, when dealing with these numbers, the processor is capable of transferring them only eight bits at a time.

6809 REGISTERS

The registers in a processor can take many forms, depending on their particular functions. Some are reserved for the internal use of the processor and cannot be accessed by the programmer. There are four main 6809 registers that the machine code programmer will use a lot.

The most commonly used register is the *accumulator*. This is where most of the data being used is stored and manipulated. For example, the usual function performed by an Assembly language ADD instruction is to add the contents of a specified memory location to the contents already stored in the accumulator. Thus, a new value will 'accumulate' in this register.

The *index register* is used to modify addresses so that we can step through tables and lists of data easily. When an instruction refers to a memory location using *indexed addressing* then the contents of this register are added on to the address given in order to specify the *effective address* of the data required. To step through a table of data, we have only to refer to the *base address* (of the first item of the table) and keep incrementing the index register. As the values stored in this register are normally addresses, index registers are generally 16 bits long, rather than eight.

The *stack pointer* is the register that indicates the location of the top of the *stack*, which is a convenient way of storing data and retrieving it quickly. The stack is used when it is necessary to save the internal contents of the processor (for example, when a subroutine is called) so that they can be restored later. The contents of some, or all, of the registers can be 'pushed onto' the stack and then 'pulled off' later when control returns to the main program. The stack pointer simply tells the processor the location of the last item put into the stack and where it can save or find the next item. Because they also refer to addresses in memory, stack pointers are generally 16 bits long.

The fourth register is very important, although its function is automatic most of the time. This is the *program counter*, which should always contain the address where the next instruction is stored. The processor goes to the location specified by the register, fetches the contents, interprets their meaning and acts on that instruction. Normally, the program counter will automatically be incremented as instructions are carried out, so that the instructions are fetched in sequence. Altering the contents of the program counter (by storing a new value there, or adding to or subtracting from the old value) will change the course of the program. In other words, this acts like a GOTO instruction, although at this level it would be referred to as a Jump (JMP) if a completely new address was provided, or a branch (BRA) if the current address was altered.

There is a fifth type of register, although it does not operate in the same way as the others. This is the *condition code* register, which is best thought of as a collection of individual bits, each representing some feature of the state of the processor. For example, one of these bits is used to signify to the processor whether the number

Popular Choice
The two most popular 6809 machines are both home computers: the Dragon 32K and 64K, and the Tandy Color Computer. There is also a wide variety of 6809 development systems in use in universities and polytechnics. The Tandy Color Computer and the two Dragon models are very similar internally, and although Dragon Data has left the computer marketplace, Tandy has agreed to provide software and technical support for Dragon users

formed as a result of an operation in one of the registers is zero. For example, we can step through a table of values by loading the total number of values into a register and subtracting one from the total every time we deal with a value. When the total reaches zero, the condition code bit tells the processor it has no more items in the table to deal with and can go on to another instruction. This type of instruction allows us to perform selection (IF statements) and looping (FOR...NEXT, WHILE...WEND, REPEAT...UNTIL statements).

Many processors maintain a *zero page*, which normally consists of the first 256 memory locations (from hex 0000 to 00FF). Values stored here can be accessed using an eight-bit address, thus making the instructions shorter and quicker to perform. The 6809 processor generalises this concept by having an eight-bit *direct page* register that provides the extra eight bits of the full address when referring to the zero page. By changing the value in this register, the zero page can be positioned anywhere in memory, or you can even have more than one of them.

A machine code program will consist of a sequence of instructions intermingled with data and addresses. Some people can actually program satisfactorily dealing directly with numeric values for all these quantities, but most of us would find this rather difficult. The Assembly language of a processor enables us to write machine code programs using much more convenient mnemonics (for instructions) and labels (for addresses and data). Thus, for example, if we need to load the data at a memory location into the accumulator we can write:

```
STORE  FCB 0
```

to reserve a place in the memory that we can refer to as STORE, in which we have temporarily placed a zero. FCB is not really an instruction, but a directive telling the program that translates Assembly language into machine code that it must substitute a particular address whenever it encounters the word STORE. Later in the program when we want the value stored there to be loaded into the accumulator we can use the instruction:

```
LDA STORE
```

which will take whatever value is stored there and load it.

Assembly language programs need translating before they can actually be run and this is the job of a program called the *assembler*. These do not need to be complicated programs because there is almost a one-to-one relationship between the Assembly language statements we write and their machine code equivalents. All that has to be done is to make the substitution and keep track of which names refer to which values or addresses.

In the next instalment, we shall study the internal structure of the 6809 processor more closely, and start to look at what instructions we have available to us.

NEW HORIZONS

Integrated Suite
Psion's Xchange is a suite of integrated business software packages based on the software created for the Sinclair QL. Xchange includes the Quill word processor; Archive database manager; Abacus financial planner and Easel business graphics system. The four packages may be purchased together or separately. Xchange is available for the IBM PC and PC XT, ACT Apricot and Apricot XI, and the Sirius I. Additional versions are planned for the Apple Macintosh and Digital Rainbow

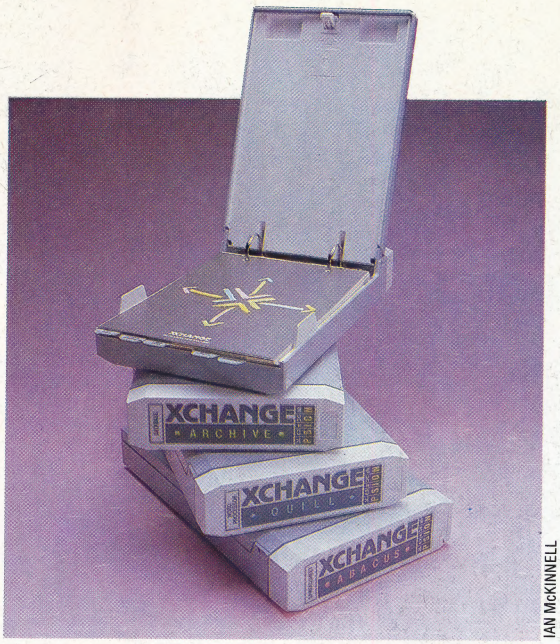
Psion is a company generally associated with Sinclair Research: it produced the Horizons program for the Spectrum, and developed the four applications programs that come with the QL. Recently, it has diversified into business software and ventured into the hardware market with its Organiser pocket computer.

Psion was founded in 1981 by Dr David Potter, a lecturer at Imperial College, London. The company's first marketing coup was a group of four packages for the newly-released ZX81: Flight Simulator, Backgammon, Vu-Calc and Vu-File, all of which were written by Charles Davies and Colly Myers. This small range of quality programs — a simulations package, a game, a spreadsheet and a database — immediately established the company's name. In 1982, when Sinclair Research came to commission a package to demonstrate the strength of the Spectrum, it was not surprising that Psion was the software house chosen to develop the cassette.

The popularity of Sinclair's machines has resulted in a large market for Psion's software. The company has had some notable successes: the combined sales of the ZX81 and Spectrum versions of Flight Simulator, for example, have exceeded 500,000 copies. Estimates for the total worldwide sales of Psion cassettes have passed the three million mark. And with its recent announcement of the 'Xchange' range of software, Psion showed its first signs of a desire to diversify — in this case by making a bid for part of the £2 million business software market.

Psion has always been an innovator. It led the way in developing the technique of 'cross-compiling' software for home computers, a process whereby a program is developed on one machine for use on another. The Horizons introductory package for the Spectrum was written on a Tandy TRS-80. Today, the company uses two VAX 750 minicomputers for all its software (see page 381). It was on these machines that the QL's suite of programs — Abacus (a spreadsheet), Archive (a database), Easel (a graphics program) and Quill (a word processor) — was written. Psion Support has organised QLUB, a helpline for QL users that guarantees a written reply to queries within 48 hours.

Plans are well advanced for a number of business programs for the IBM PC, Apricot, Sirius, DEC Rainbow and Macintosh computers. Like the QL suite, the Xchange range of programs features a spreadsheet, database, graphics program and word processor. What sets these apart from other business packages, Psion claims, is the ease with which data can be transferred between them.



IAN McKINNELL

Another development from Psion that has gained a lot of attention is the Organiser (see page 441). The appearance of a pocket computer from what had been regarded as a software company came as a surprise to many people. A company spokesman, Robin Kinnear, puts the development in perspective: 'The key is that Psion *is* a microcomputer software company. We thought of the Organiser as a very smart idea in terms of software *packaging* and looked around and found there was nothing comparable. So Psion decided to make its own hardware. The development was very much software-led.'

At the moment, only three applications packages (science, maths and finance) are available for the Organiser, along with the eight Kbyte and 16 Kbyte RAM packs (or 'datapaks' as they are called). Psion is considering adding other program packs, and it has also been approached by a variety of individuals who wish to write software for the machine.

Market expansion is another company priority at the moment: subsidiaries have recently been established in the United States and South Africa, and contracts for extensive distribution of Psion products throughout Europe have been signed. Furthermore, Sinclair Research has launched a big sales drive in Eastern Europe, beginning with the export of 400 ZX81s to Czechoslovakia. Psion, which has already taken some trouble to produce foreign language versions of its software, is sure to follow.

Leading Light
Dr David Potter is the founder and majority shareholder of Psion. Formerly an academic who specialised in Computational Physics at Imperial College, London, and the University of California, he founded Psion as Potter Scientific Investments

Overall control of ClubSpot is now vested in the Association of Computer Clubs' Communication Committee. This committee is responsible for a general overview of the database, the organisation of conferences to train new editors and running a ClubSpot stand at computer exhibitions.

The manager of ClubSpot is Len Stuart. Together with his deputy, Vernon Quaintance, and several assistant managers, it is his job to look after the day-to-day running of the database. The work of the managers is to create and delete frames, deal with all the messages sent to ClubSpot and perform certain other tasks, such as designing new parts of the database.

The people who actually do the work are the editors. Mostly computer club members, they use ordinary home micros to place information onto the system. Many ClubSpot editors use BBC Micros, though others use Spectrums, Apples and Commodore machines. ClubSpot editors like to boast that although they are amateurs producing a database with home micros, theirs is often better than that produced by professional companies.

Prestel is structured so that there are six GEC mainframe computers open to the public: two in Birmingham and four in London. 'Dickens' and 'Keats' are the Birmingham mainframes, 'Kipling', 'Dryden', 'Derwent' and 'Enterprise' are their counterparts in the capital. Ninety-six per cent of the country can access one of these for a local phone call charge. To put information onto the system, Information Providers must phone up a seventh computer, situated in London, called 'Duke'.

When a ClubSpot editor has entered his ClubSpot identity and password, he then calls up page 910 and enters a further password to obtain the Prestel editing system. The two main commands available at this stage are a and o: a allows a frame to be 'amended' while o ('overwrite') allows the editor to define the 'routes' leading from the frame, as well as certain other details, such as whether the frame can be accessed by the general public.

Several editors have written programs that allow them to design whole frames on their home micro, which can then be 'uploaded'. This can turn a 10 minute phone call into a two minute one.

To qualify to become a ClubSpot editor you must prove that you are capable of using the system (usually by having attended an Editors' Training Conference) and have the authority to represent a club. Editors must undertake to keep their material within acceptable limits. At the present time there are over 30 editors actively updating the pages of ClubSpot.

ClubSpot is not self-financing; all of its facilities are donated by Prestel. A Main Information Provider outside ClubSpot would have to pay over £5,000 for a three-digit front page, and meet the running charges and any extra facilities. By providing these facilities free of charge, Prestel clearly shows that it attaches considerable value to ClubSpot. Micronet also makes a significant contribution to the running costs of the database.

The future of ClubSpot looks very bright. It has proved an excellent testing ground for new ideas and this is expected to continue. It can be seen as an example of what the enthusiastic amateur can do with a home micro. In many cases, the material in ClubSpot is better than that produced by professionals on much more expensive equipment.

In the not too distant future, people will be able to use their micro to do a multitude of tasks from the comfort of their own home. They will be able to work, shop and communicate over large distances. They should not be surprised to find that amateur input will provide a major influence on the development of computer communications.

IMAGES PRODUCED ON THE ARTRON 2000 STUDIO COMPUTER; COURTESY OF GRAPHIC PRODUCTS, LONDON (EXCLUSIVE DISTRIBUTOR)